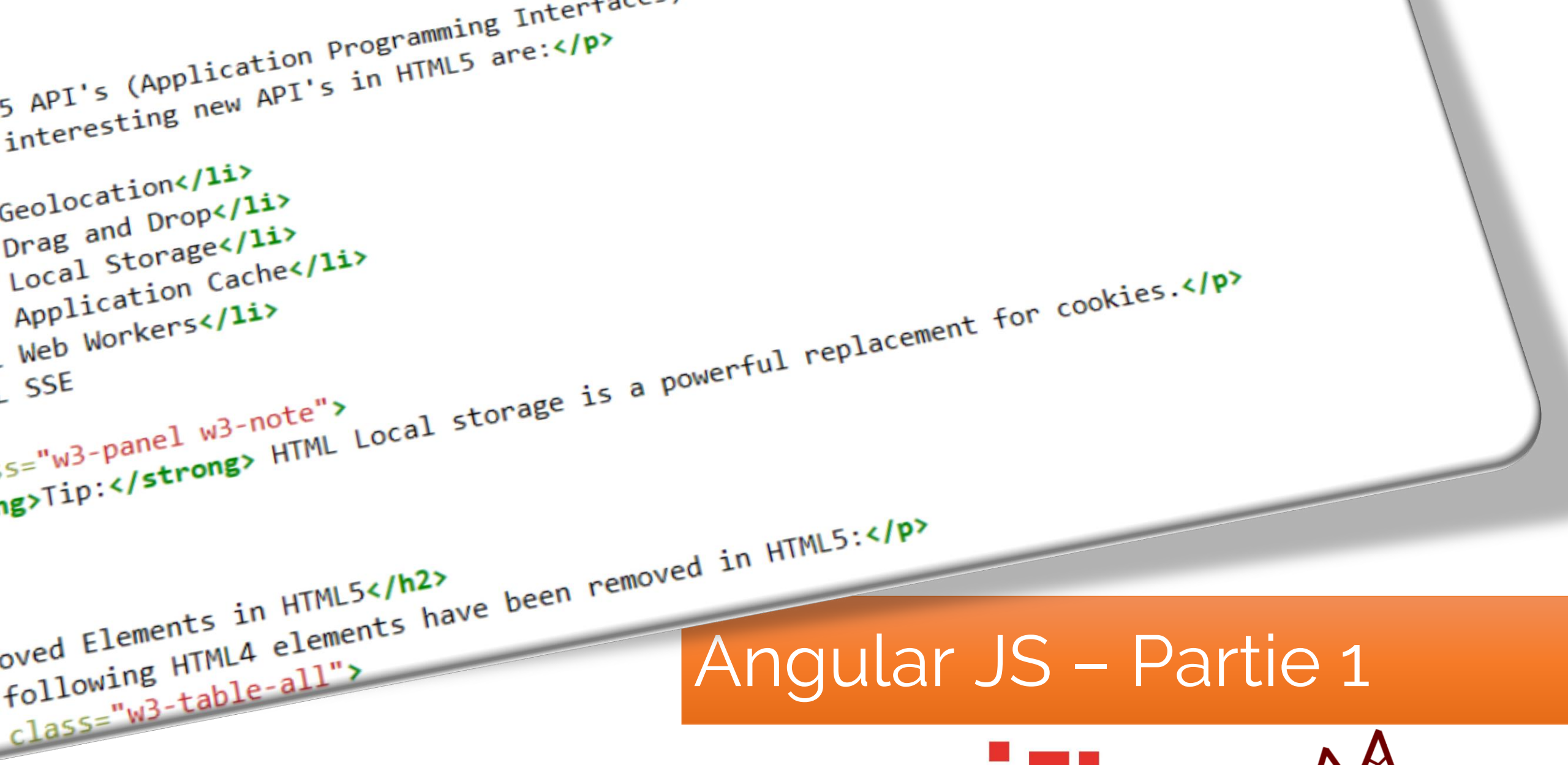
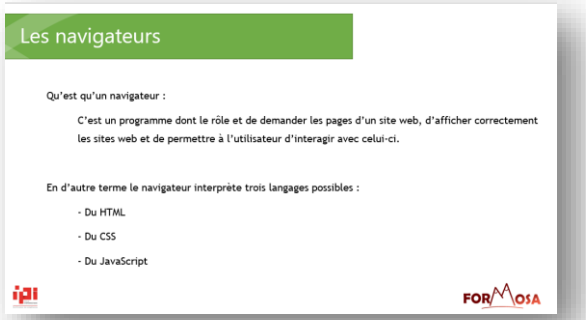


Angular JS – Partie 1

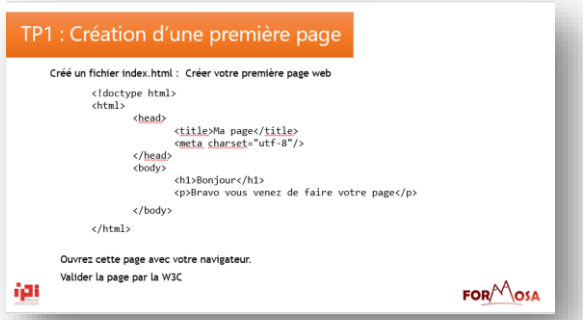
ÉCOLE D'INFORMATIQUE
Certificateur de compétences

FORMOSA

Notations



Les slides avec un bandeau vert représentent les parties plus théoriques.



Les slides avec un bandeaux Orange représente les TP

Bandeaux

Les bandeaux bleu sont utilisé pour les portions de code

```
Portions de code
```

Notes Informative

Notes importante

Notes très importante

Dans ce document

Le plan de cette formation :

Présentation d'AngularJS

Qu'est ce que AngularJS ?

Angular JS est un framework permettant la réalisation d'application en JavaScript de manière plus rapide.

Ce framework a été l'un des plus populaire en version 1.

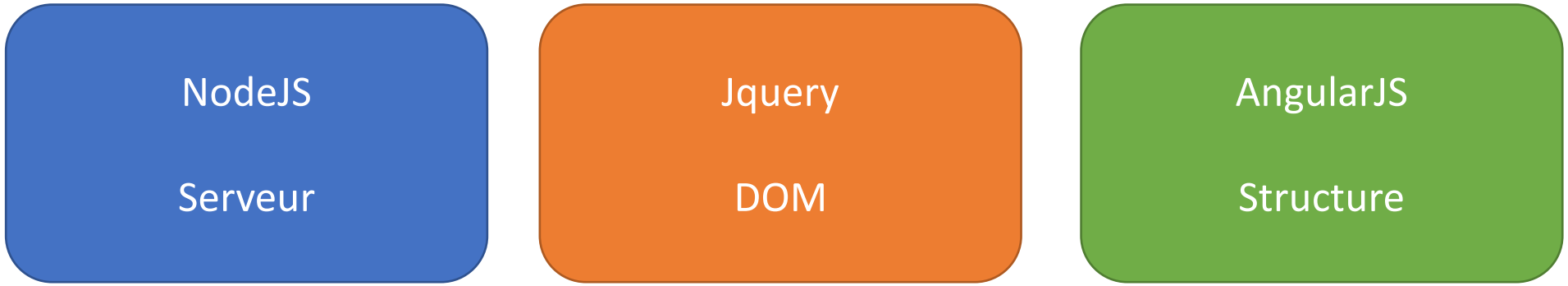
AngularJS est maintenant en version 4.

De nombreuses entreprise refondent maintenant leur SI pour intégrer AngularJS à la place des JSF et des JSP.

Les problématiques

Le calcul des pages coté serveur est devenu trop long. De nos jour nous souhaitons pouvoir utiliser la puissance de calcul du Client.

Le modèle MVC (Model View Controller) est une bonne approche cependant très peu de framework l'implémente complètement.



La Solution d'AngularJS & Philosophie

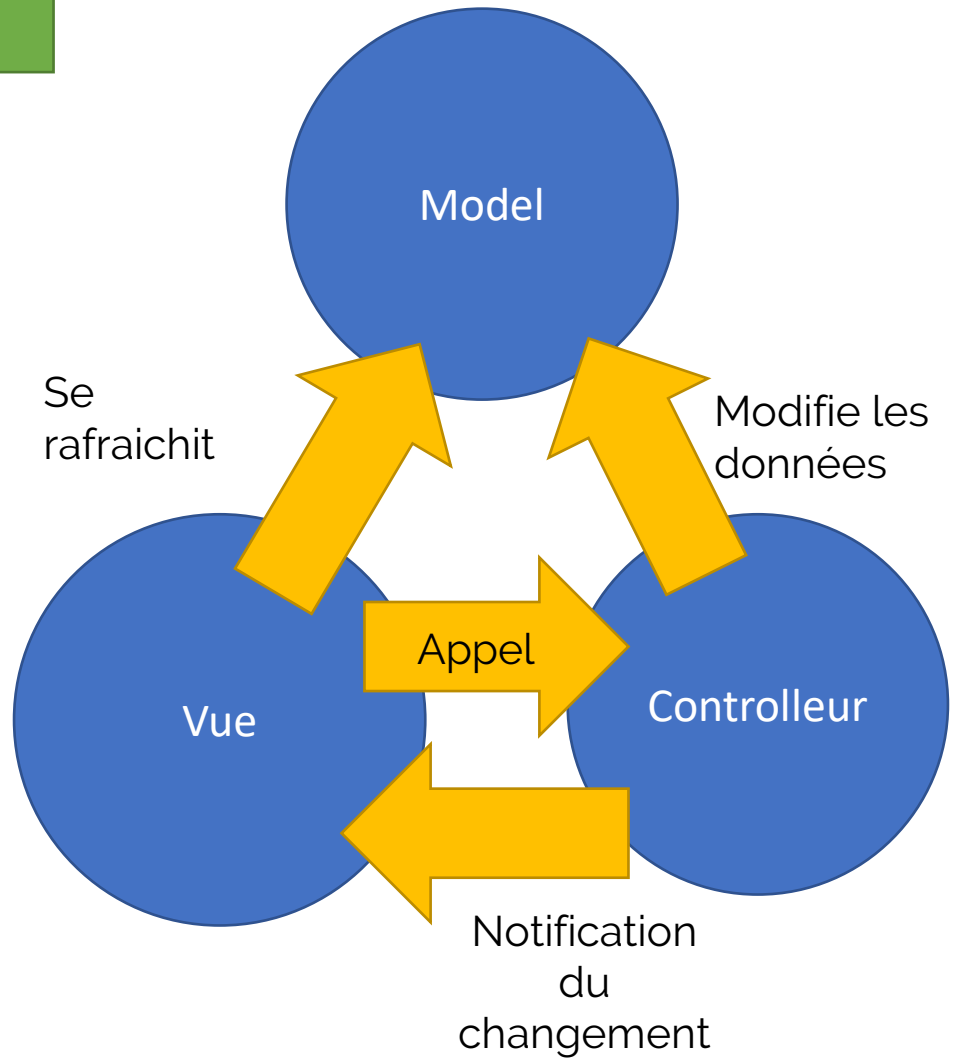
AngularJS offre :

- Rapidité d'implémentation
- Modélisation en MVC
- Un niveau d'abstraction agréable

La modification du modèle mets à jour la vue instantanément.

Angular vise à donner une modélisation correct d'une application

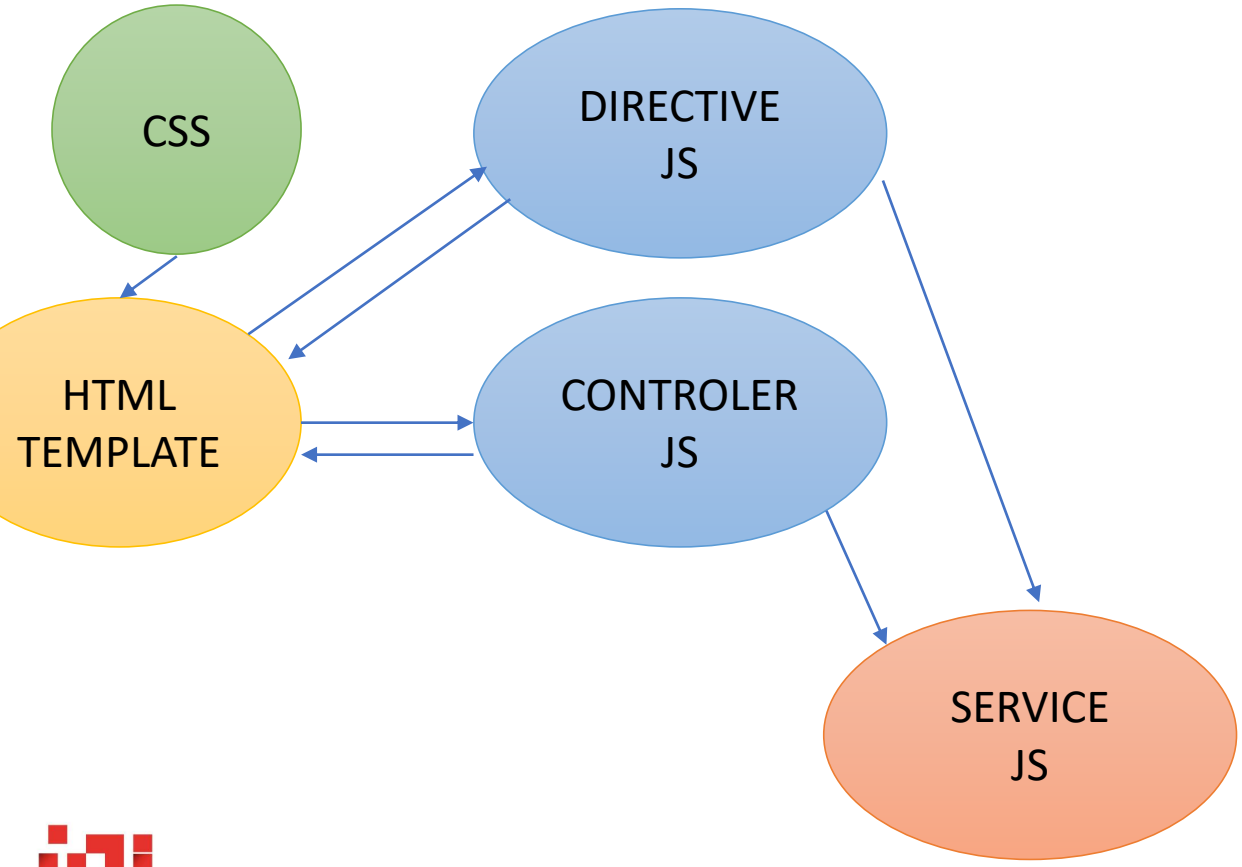
Où il n'est pas nécessaire de modifier la vue mais plutôt de modifier les données.



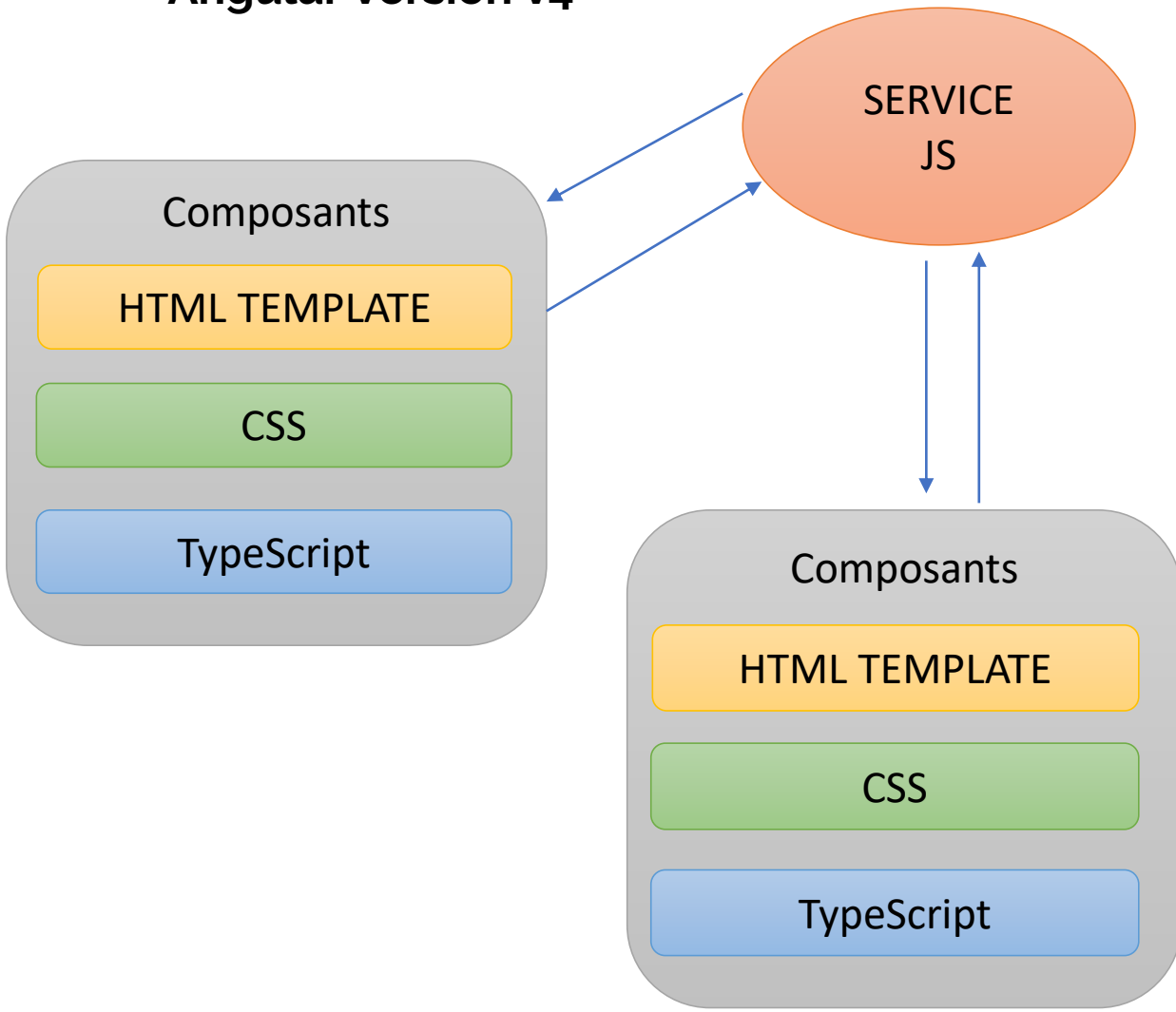
En Angular nous n'accédons donc pas à l'arbre DOM !

Évolution du Framework

Angular version v1



Angular version v4



Évolution du Paradigme

Angular version v1

Programmation Fonctionnelle JS

Langage Faiblement Typé JS

Angular version v4

Programmation Objet et Composants

Langage Fortement Typé TypeScript

Qu'est ce que le TypeScript ?

Le TypeScript est décrit comme une surcouche du JavaScript

Ce langage n'est pas encore interprété nativement par le navigateur plutôt il va générer du JavaScript

Il se repose sur un Paradigme Programmation Orientée Objets

Il offre rigidité, Structure et typage au JavaScript

TypeScript

Qu'est ce que JavaScript ?

Javascript est une implémentation du Standard ECMA Script (ES5)

Les versions utilisés ces dernières années sont ES5

Ce Standard à finalement évolué avec l'apparition d'applications modernes et l'évolution des besoins :

- le standard ES6 ou ECMA 2015 est l'évolution de ES5

Qu'est ce que ES2015

- ES2015 = ES6 = ES5 + Quelques fonctionnalités en plus

Qu'est ce que TypeScript ?

Typescript est un langage (en quelques sorte l'évolution de Javascript) satisfaisant le standard d'ES6

Il ajoute la notion de Type à JavaScript

TypeScript est tranformé en JavaScript après le processus de compilation

Il offre plus de rigidité et plus de sécurité à JavaScript

Installer le Transpiler TypeScript

Pour installer le Transpiler TypeScript exécuter la commande :

Installer TypeScript

```
npm install -g typescript
```

Créer un son premier fichier TypeScript

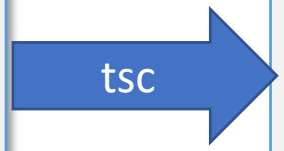
Le processus de compilation se fait en appelant la commande :

```
tsc 01_sample.ts
```

```
function bonjour(personne) {
    return "Bonjour, " + personne;
}

var nom = "Guillaume";

document.body.innerHTML = bonjour(nom);
```




```
function bonjour(personne) {
    return "Bonjour, " + personne;
}

var nom = "Guillaume";

document.body.innerHTML = bonjour(nom);
```

01_sample.ts

01_sample.js

Créer un son premier fichier TypeScript

Le processus de compilation se fait en appelant la commande :

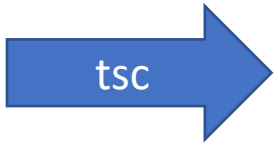
```
tsc 02_sample.ts
```

```
function bonjour(personne:string) {
    return "Bonjour, " + personne;
}

var nom = "Guillaume";

document.body.innerHTML = bonjour(nom);
```

02_sample.ts




```
function bonjour(personne) {
    return "Bonjour, " + personne;
}

var nom = "Guillaume";

document.body.innerHTML = bonjour(nom);
```

02_sample.js

Créer un son premier fichier TypeScript

Le processus de compilation se fait en appelant la commande :

```
tsc 03_sample.ts
```

```
function bonjour(personne:string) {
    return "Bonjour, " + personne;
}

var nom = 123;

document.body.innerHTML = bonjour(nom);
```

tsc

```
function bonjour(personne) {
    return "Bonjour, " + personne;
}

var nom = 123;

document.body.innerHTML = bonjour(nom);
```

03_sample.ts

03_sample.js

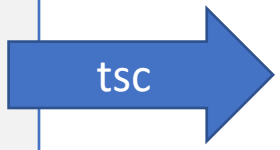
"Argument of type 'number' is not assignable to parameter of type 'String'."

Les Types Typescript

Il existe de nouveau TypeScript :

Chaine de Caractère	`var nom:string = "Guillaume";`
Nombre	`var age:number = 123;`
Booléen	`var fait:boolean = false;`
Tableau	`var list:number[] = [1, 2, 3];`

```
function bonjour(personne) {
    return "Bonjour, " + personne;
}
var nom:string = "Guillaume";
var age:number = 123;
var fait:boolean = false;
var list:number[] = [1, 2, 3];
document.body.innerHTML = bonjour(nom);
```




```
function bonjour(personne) {
    return "Bonjour, " + personne;
}
var nom = "Guillaume";
var age = 123;
var fait = false;
var list = [1, 2, 3];
document.body.innerHTML = bonjour(nom);
```

Les Types Typescript

```
class Stylo{
        encre: number;
        color: string;

    constructor(color:string) {
        this.encre = 100;
        this.color = color;
    }

    changeCouleur(color:string){
        this.color=color;
    }
}
var s1:Stylo = new Stylo("red");
s1.changeCouleur("blue");
console.log(s1);
```

TypeScript introduit avant tout la notion de classes propre à la programmation orientée objet :

```
{ encre: 100, color: "blue" }
```

Démarrage Rapide

Installer l'environnement

Pour utiliser AngularJS nous devons installer :

- NodeJS avec la commande NPM

- Un éditeur de type bracket.io ou notepad est suffisant.

Créer un nouveau projet

Pour créer un nouveau projet, il faut exécuter la commande suivante :

Installer Angular Cli

```
npm install –g @angular/cli
```

Créer un projet

```
ng new nouvelle-app
```

Servir une application

Il faut maintenant lancer le serveur et servir le site :

Installer Angular Cli

```
Se positionner dans le répertoire de l'application :

ng serve --open
```

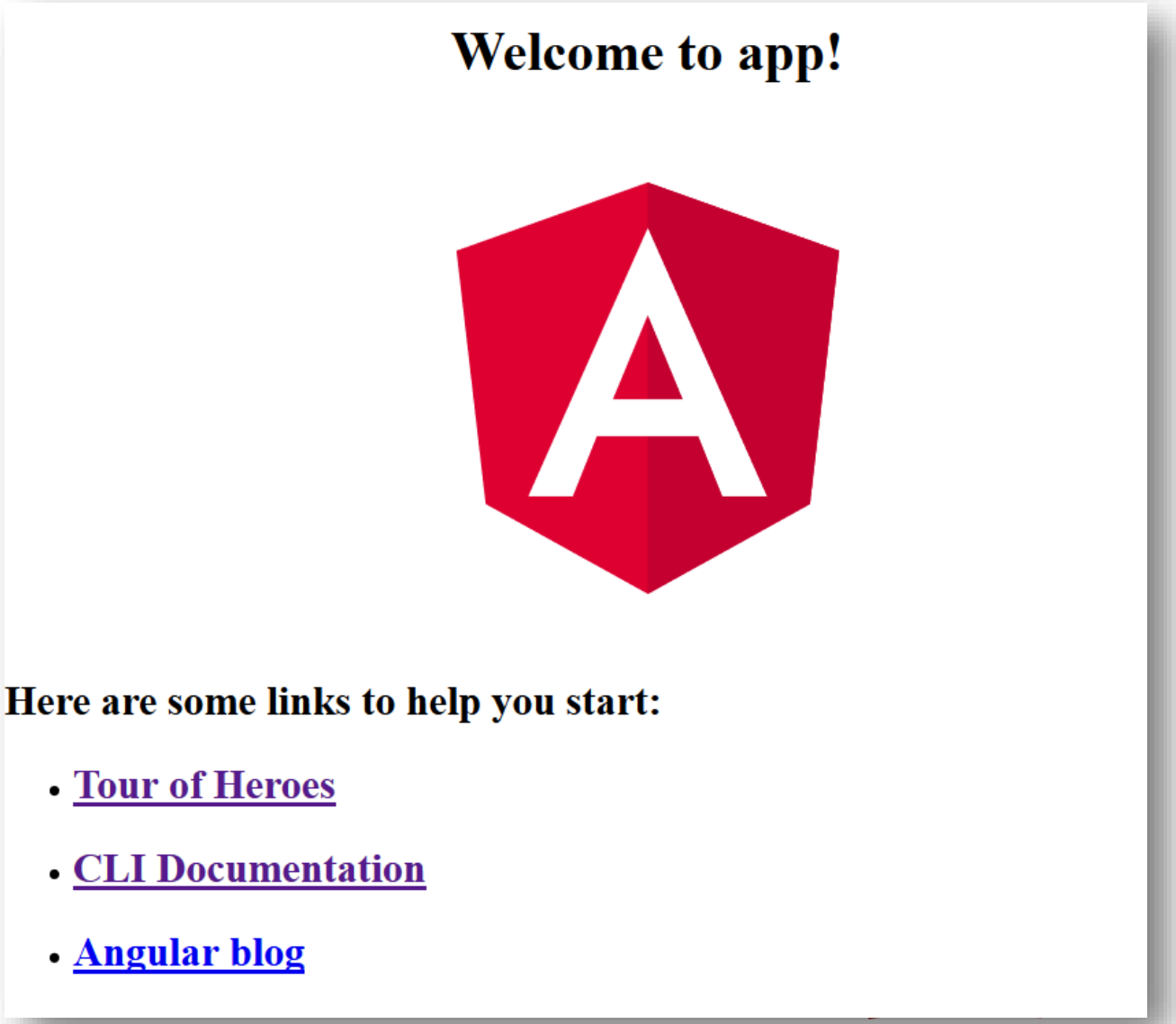
Modification du projet

Angular a sorti un nouveau concept : le composant.
Angular cli à créer 4 fichier définissant ce composant :

- le fichier ts : typescript
- le fichier css : la feuille de style
- le fichier html : la structure
- le fichier spec.ts

Aller donc dans le fichier : app.component.html, app.component.css pour modifier le contenu du composant.

A la modification le serveur se remet à jour automatiquement



Composition d'un projet

app/app.component.{ts,html,css,spec.ts}

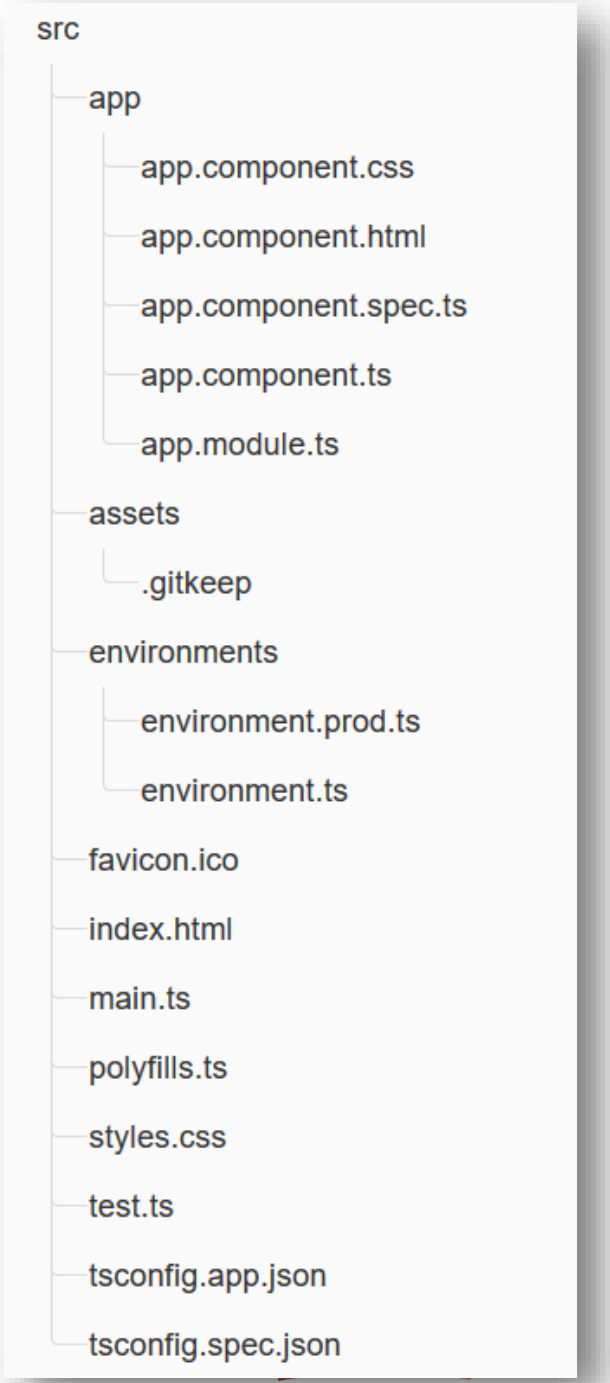
Définit AppComponent, le composant App avec un template HTML, une feuille de Style, un fichier TS pour définir sa structure de données, un fichier spec.ts pour les tests unitaires.

app/app.module.ts

Définit AppModule qui dit à angular comment assembler l'application. Au démarrage ce composant ne déclare qu'un seul composant.

app/assets

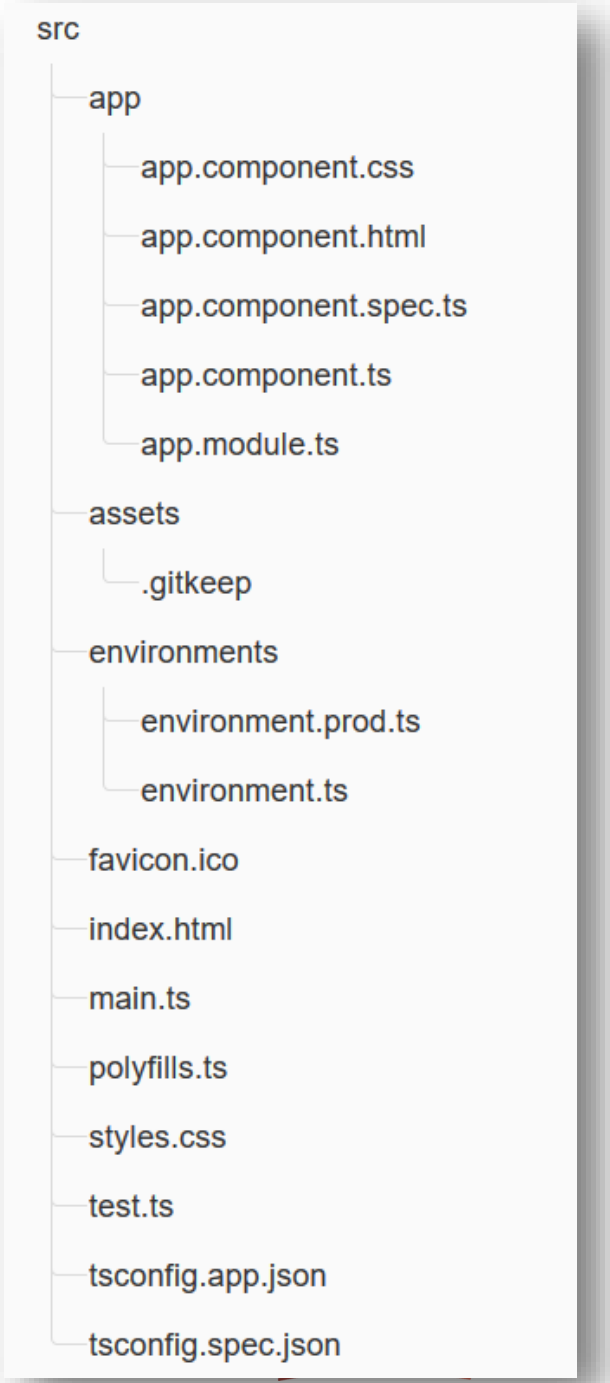
Stocker des images ou n'importe quel fichier à être copier quand l'application Build



Composition d'un projet

Environments/*

Contient les variables d'environnement pour les différents environnement d'exécution : Prod, Recette, Intégration etc.



Concept de Composant sous AngularJS

Le Fichier TypeScript

Le fichier TypeScript va se décomposer en différentes sections :

- les **importations** qui sont soit des importations de bibliothèque Angular soit des importations des bibliothèques personnelle du développeur.

- Un **ensemble de classes** internes au composants que le développeur définit

- La **déclaration du composant** par l'annotation @Component permettant de définir :
    - le sélecteur
    - le template HTML
    - la feuille de style CSS
    - etc.

- La **classe principale** du composant

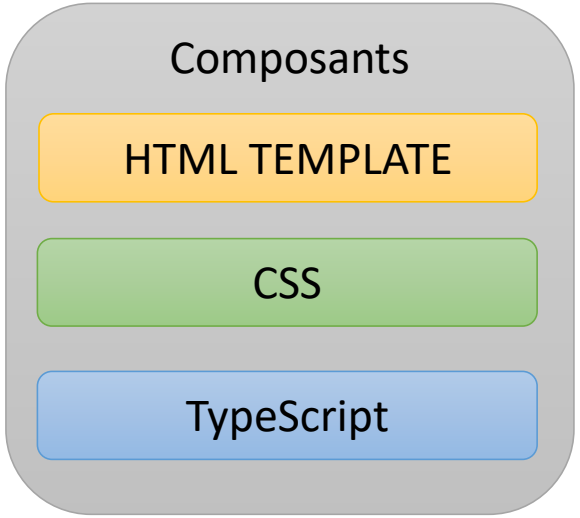
Structure d'un composants

Un composants se compose :

- D'un fichier HTML

- D'un fichier CSS

- D'un fichier TypeScript



La convention de nommage :

```
[NomDuComposant].component.html
[NomDuComposant].component.css
[NomDuComposant].component.ts
```

Selecteur

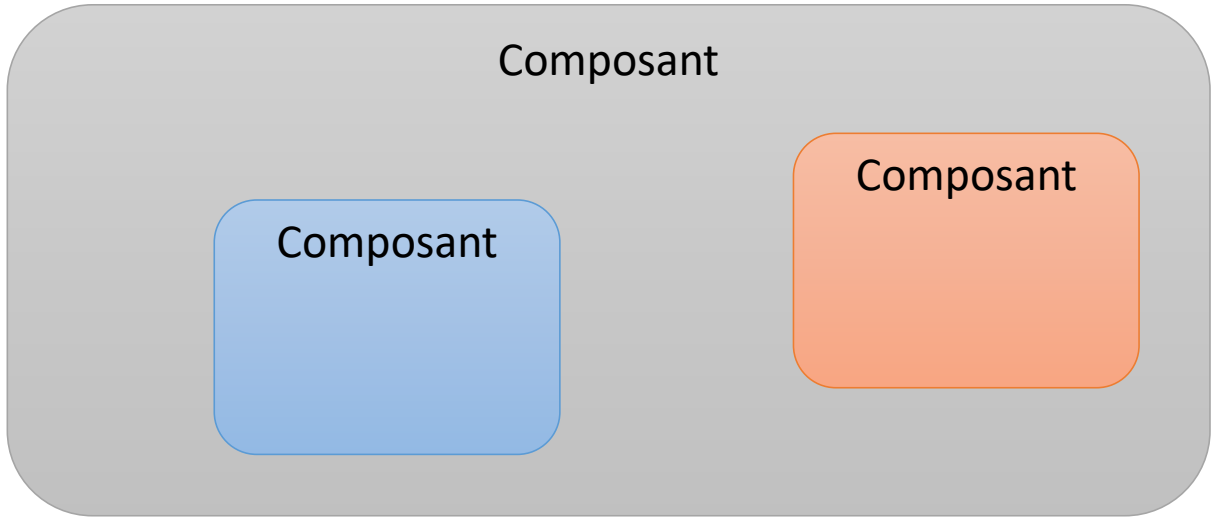
Un composants va définir un sélecteur, en d'autre terme c'est la balise HTML qui va permettre de positionner notre composant dans notre template HTML

Dans le cas de composants Racine d'une page, il n'est pas nécessaire de mettre le sélecteur dans le template HTML.

Composition/Imbrication des Composants

Le principe d'un composant et cela est valable dans tous les framework de développement d'interface graphique.

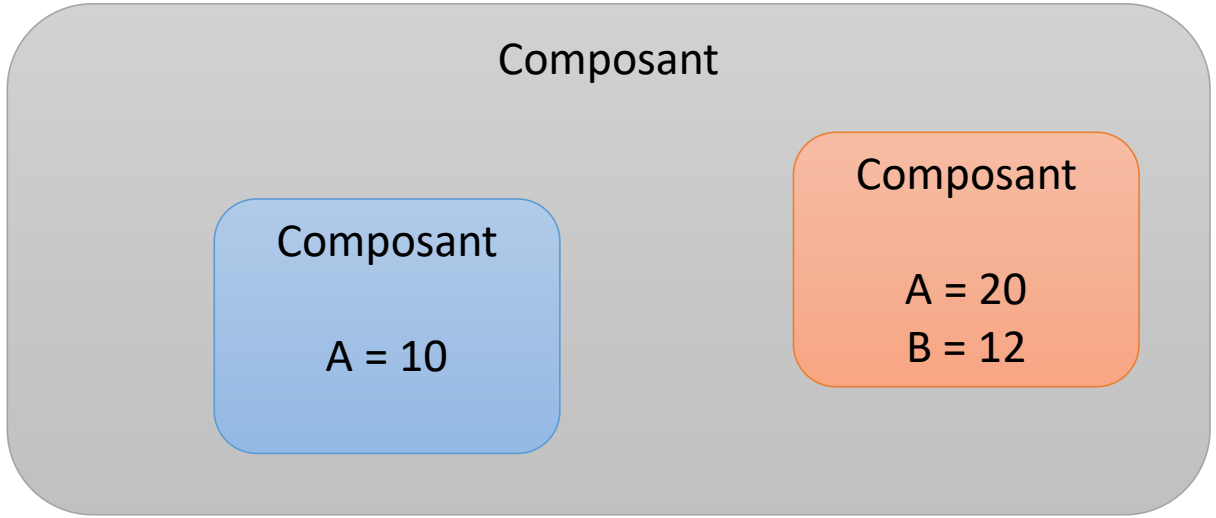
Est qu'un composant peut s'imbriquer dans un autre composant :



Isolation des composants

Un composant est isolé des autres composants. Ces variable et attributs lui sont propre.

Il est cependant paramétrable.

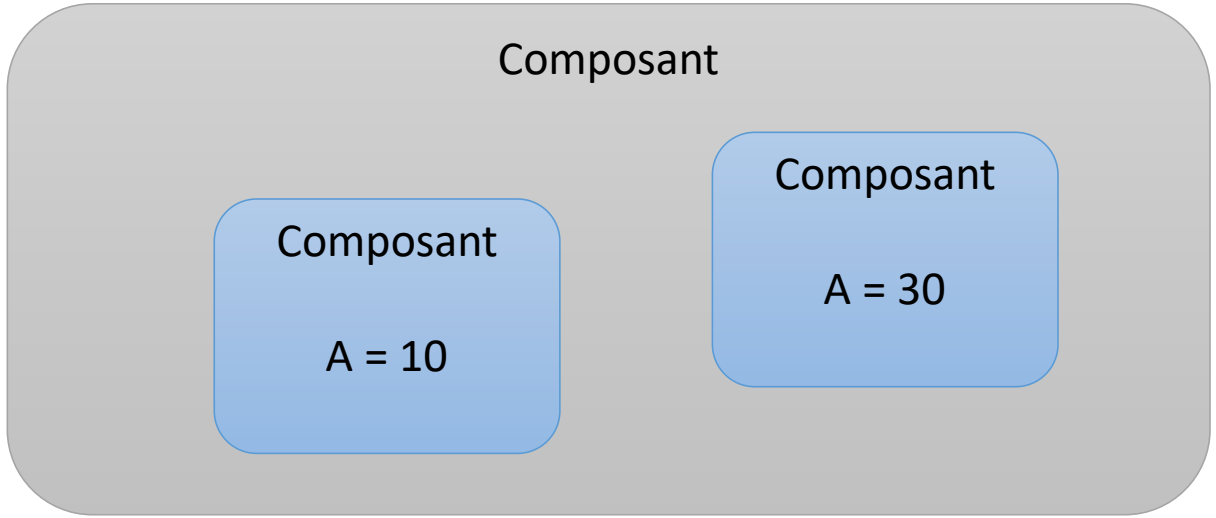


Duplication des composants

Un composant est duplicable.

Sans modification du code du composant, un composant doit pouvoir être duplicable et avoir un comportement identique à l'autre composant

Bien sur il demeure toujours isolé de l'autre composant.



Résultat

ComponentA.component.css

```
* {
        background-color: lightblue;
}
```

ComponentB.component.css

```
* {
        background-color: lightcoral;
        margin :10px;
}
```

app.component.html

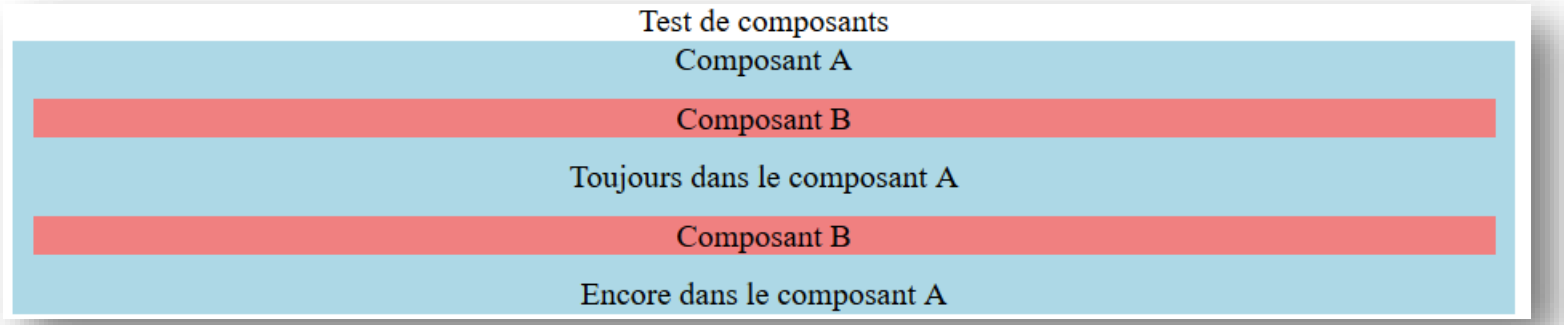
```
<div style="text-align:center">
    Test de composants
    <compoa></compoa>
</div>
```

ComponentB.component.html

```
<div style="text-align:center">
   Composant B
</div>
```

ComponentA.component.html

```
<div style="text-align:center">
    Composant A
    <compob></compob>
    Toujours dans le composant A
    <compob></compob>
    Encore dans le composant A
</div>
```



Résultat

Binding et Classes

1-way binding

Binding signifie en Anglais "Associer" ou "lier".

Le Binding sous angularJS fait le lien entre un élément Graphique et un élément du modèle de données.

Le marqueur `{{nomVariable}}` permet d'établir ce binding

```
'<h1>{{title}}</h1>'
```

Si `title = "Détail personne"` nous donnera :

Afficher le détail d'une personne

```
import { Component } from '@angular/core';

@Component({
  selector: 'app-root',
  template: '<h1>{{title}}</h1><h2>Détail sur {{prenom}} {{nom}} </h2>',
  styleUrls: ['./app.component.css']
})

export class AppComponent {
  title = 'Détail personne';
  nom = 'Kent';
  prenom = 'Clark';
}
```

Importation

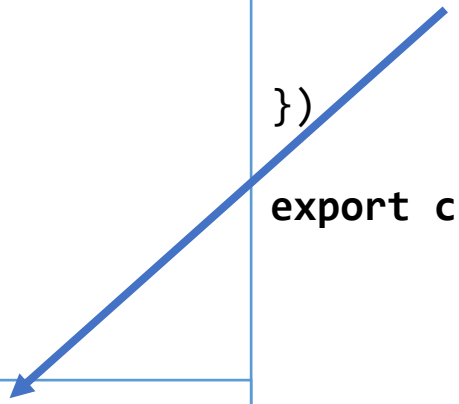
Déclaration du composant

Classe du composant

Il est possible d'utiliser templateUrl pour spécifier le template dans un fichier

Séparer le template

Séparer le template est recommandé pour les gros composants ou pour avoir un code bien structuré.



```
<h1>{{title}} </h1>

<h2>Détail sur {{prenom}} {{nom}}
</h2>
<p><b>Description : </b></p>
<p>{{description}}</p>
```

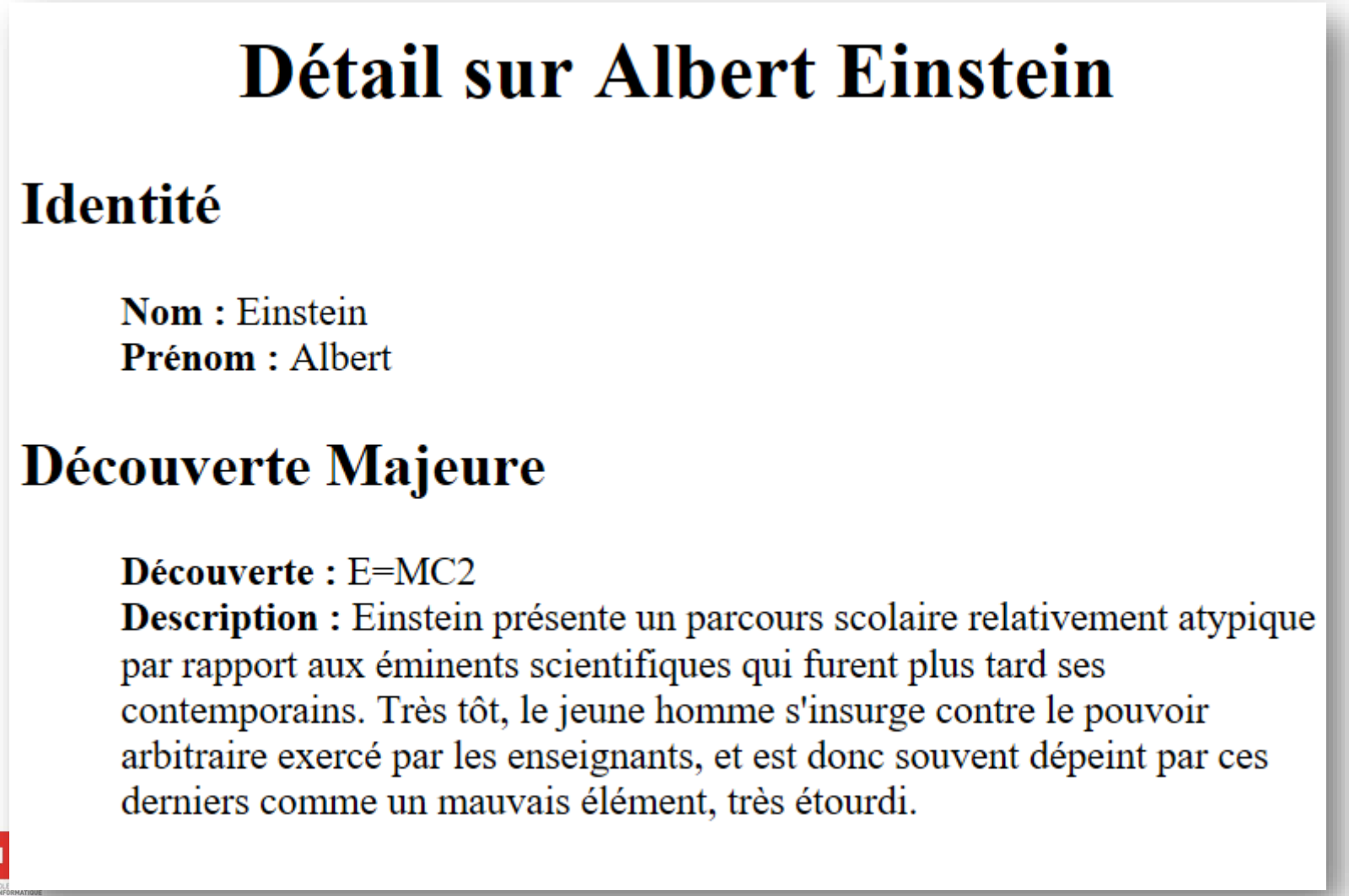
app.component.html

```
import { Component } from '@angular/core‘;
@Component({
        selector: 'app-root',
        templateUrl: './app.component.html',
        styleUrls: ['./app.component.css']
})

export class Appcomponent {
        title = 'Détail personne';
        nom = 'Kent'
        prenom = 'Clark'
        description= 'Clark Joseph Kent is a fictional
        character appearing in American comic books
        published by DC Comics. Created by Jerry Siegel
        and Joe Shuster, he debuted in Action Comics #1
        (June 1938) and serves as the civilian and
        secret identity of the superhero Superman. '
}
```

app.component.ts

Réaliser cette page



Détail sur Albert Einstein

Identité

Nom : Einstein
Prénom : Albert

Découverte Majeure

Découverte : E=MC2
Description : Einstein présente un parcours scolaire relativement atypique par rapport aux éminents scientifiques qui furent plus tard ses contemporains. Très tôt, le jeune homme s'insurge contre le pouvoir arbitraire exercé par les enseignants, et est donc souvent dépeint par ces derniers comme un mauvais élément, très étourdi.

```
title = 'Détail sur '
nom = 'Einstein'
prenom = 'Albert'

majorWork = 'E=MC2'
description = "Einstein
présente un parcours
scolaire relativement
atypique par rapport aux
éminents scientifiques qui
furent plus tard ses
contemporains...
```

Définition de classes et création d'un Objet

Nous pouvons créer des objets composés de différents champs.

C'est objets sont utilisable dans le template,

```
import { Component } from '@angular/core‘;

@Component({
        selector: 'app-root',
        templateUrl: './app.component.html',
        styleUrls: ['./app.component.css']
})

export class AppComponent {
        title = 'Détail personne';
        personne = {
                nom : 'Kent',
                prenom : 'Clark',
                description: 'Clark … Superman.'}
}
```

Création de l'objet

app.component.ts

```
<h1>{{title}}</h1>
<h2>Détail sur {{personne.prenom}}
{{personne.nom}} </h2>
<p><b>Description : </b></p>
<p>{{personne.description}}</p>
```

app.component.html

Modification du Template HTML

2-way binding

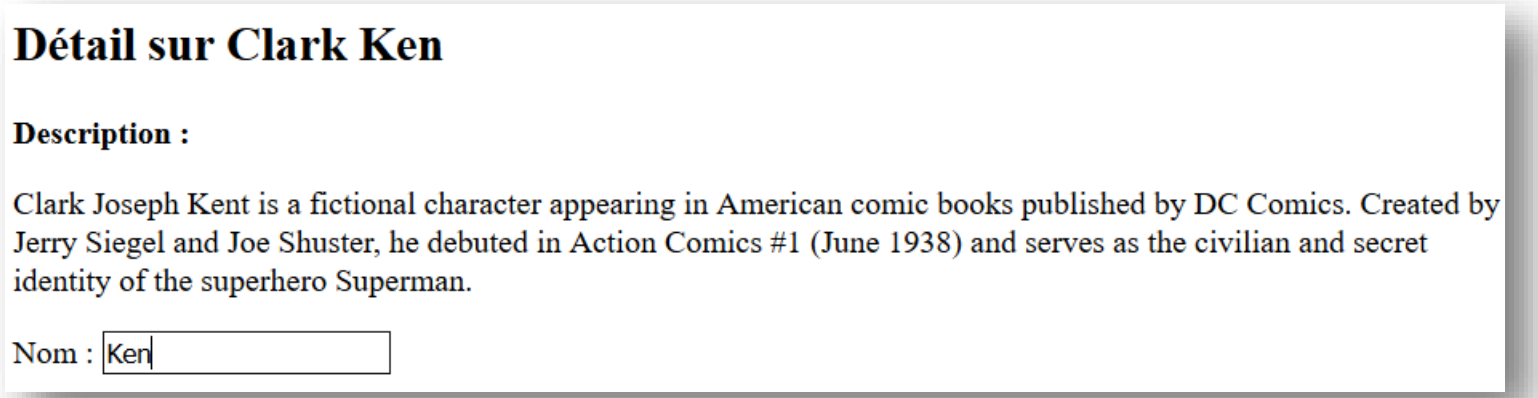
Nous appelons 1-way binding, le binding en lecture.

Nous appelons 2-way binding, le binding en lecture et écriture.

Le 2-way binding est donc principalement utilisé pour la création de formulaire

```
<input [(ngModel)]="personne.nom" placeholder="Nom">
```

Le principe est de modifier dynamiquement une variable sans le rafraichissement de la page :



2-way binding

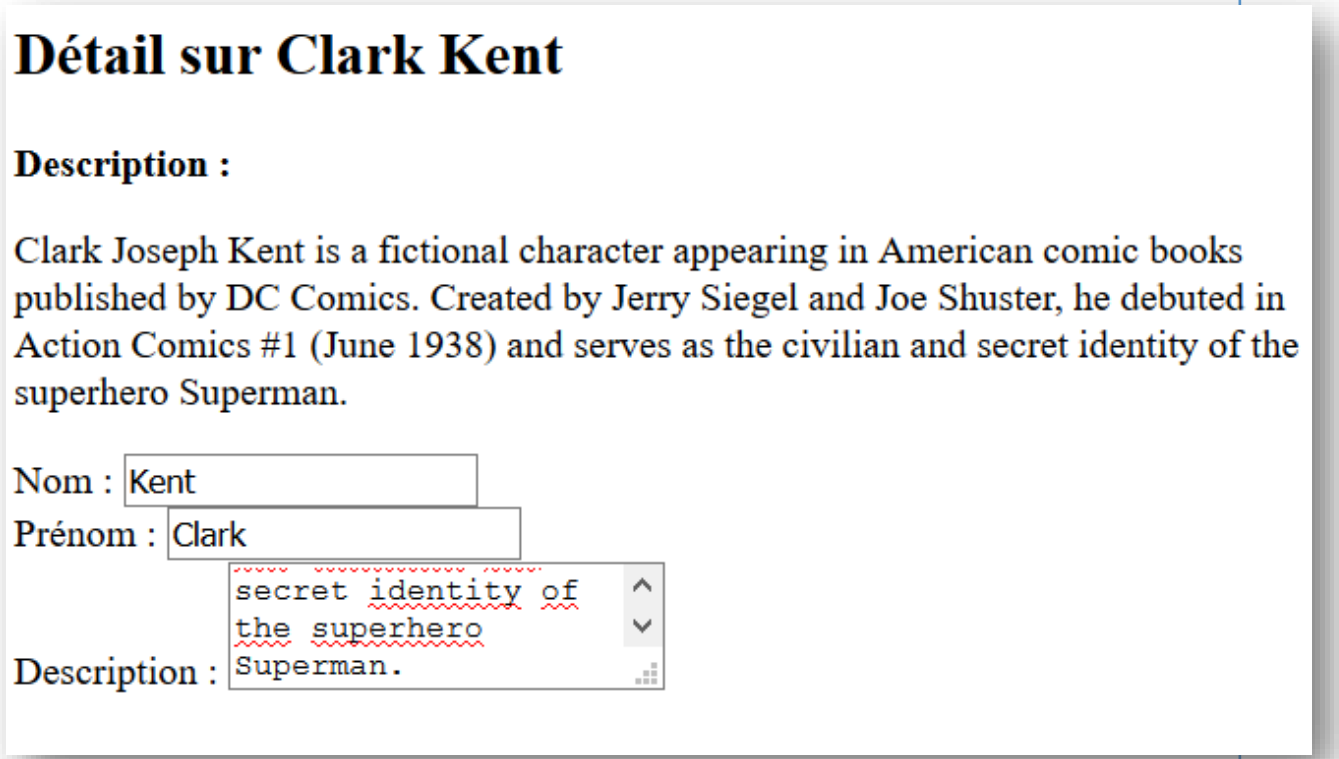
```
import { BrowserModule } from '@angular/platform-browser';
import { NgModule } from '@angular/core';
import { FormsModule }   from '@angular/forms';

import { AppComponent } from './app.component';

@NgModule({
  declarations: [
    AppComponent
  ],
  imports: [
    BrowserModule, FormsModule
  ],
  providers: [],
  bootstrap: [AppComponent]
})
export class AppModule { }
```

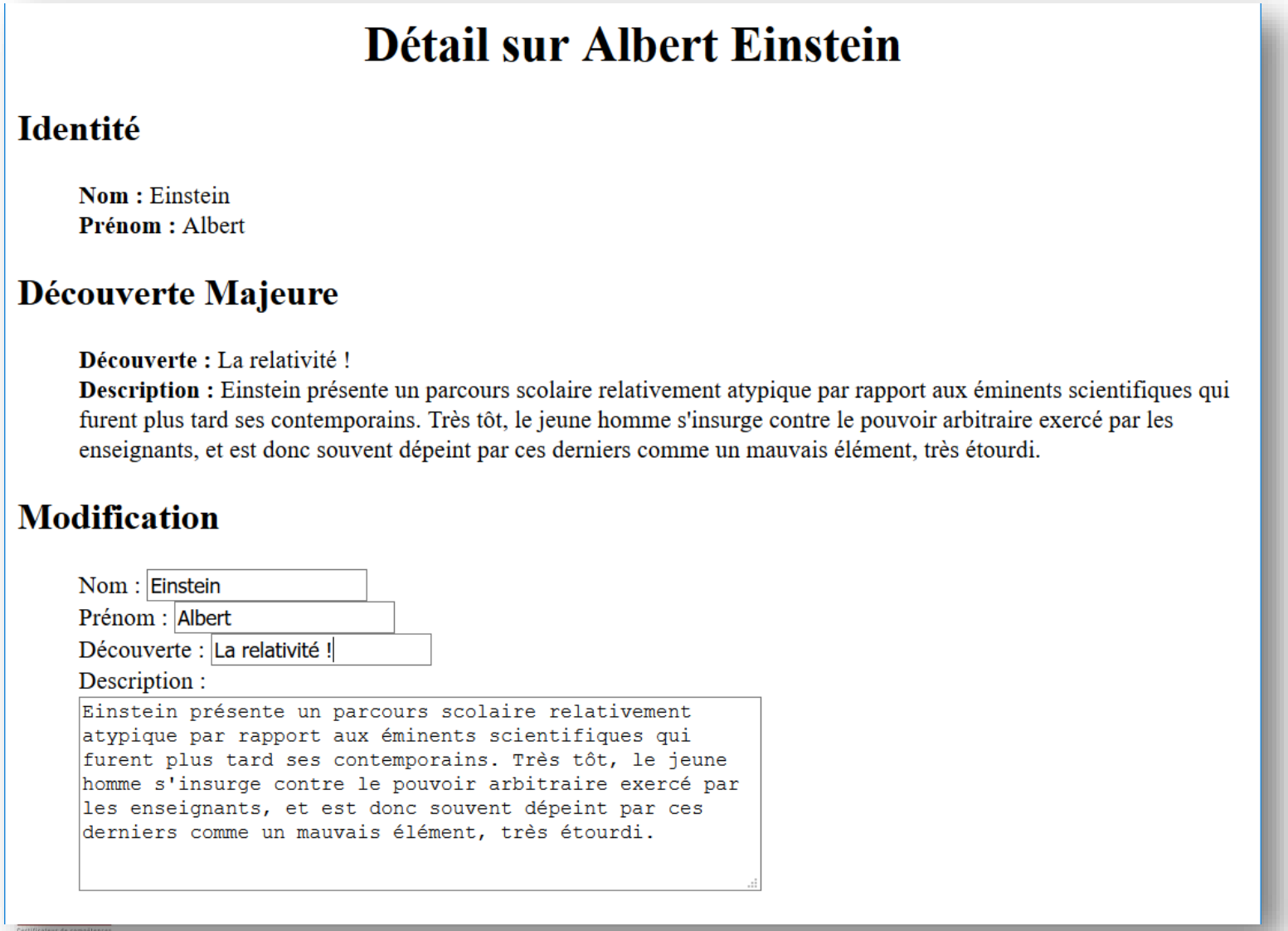
app.module.ts

Ajouter le prénom et la description




```
<div>
        <div>
            <label>Nom : </label>
            <input [(ngModel)]="personne.nom"
                        placeholder="Nom"/>
        </div>
        <div>
            <label>Prénom : </label>
            <input [(ngModel)]="personne.prenom"
                   placeholder="Prenom"/>
        </div>
        <div>
            <label>Description : </label>
            <textarea
                [(ngModel)]="personne.description"
                placeholder="Description">
            </textarea>
        </div>
    </div>
```

Ajouter un formulaire d'édition



Ajouter un formulaire d'édition permettant de modifier la personne.

La modification est effectuée en temps réel.

FORMOSA

2-Way Binding

Expérimentez le binding sur d'autres éléments que des champs textes

- Elements Radio
- Elements CheckBox
- Choix multiples
- etc.

Evénement

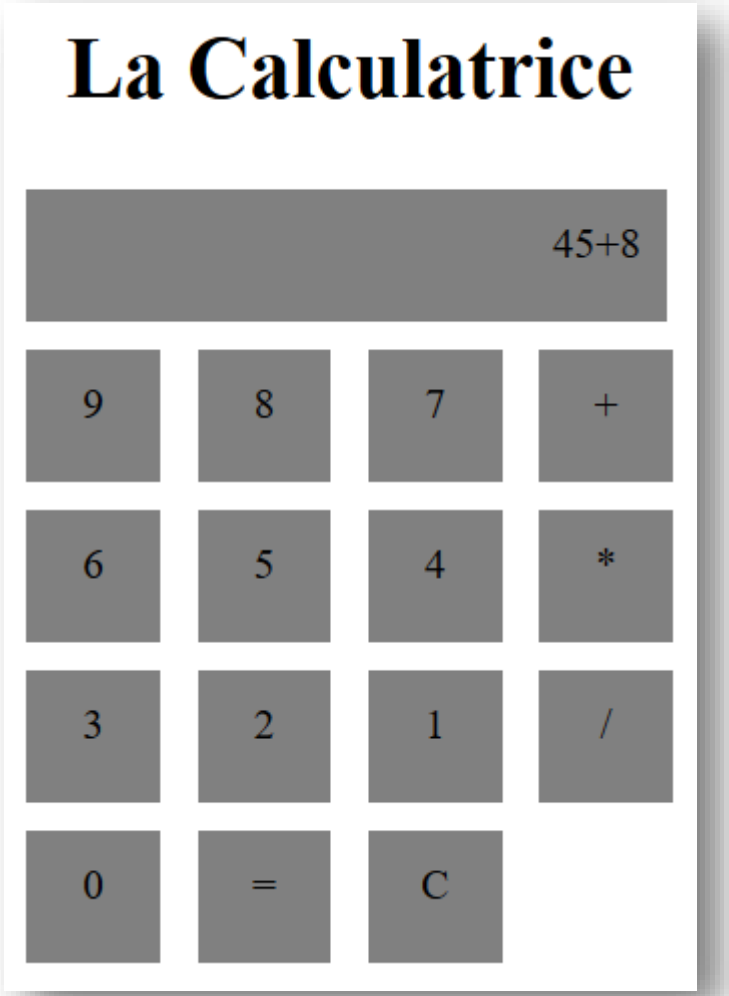
La gestion des événement s'effectue avec les attributs spéciaux comme l'attributs (Click)

```
<ul>
    <li (click)="clickOn()">Element1</li>
    <li (click)="clickOn()">Element2</li>
</ul>
```

```
export class AppComponent {
title2 = 'app3';
abc = 'salut';

clickOn(){
        console.log("Salut");
}}
```

Implémenter la Calculatrice



Implémenter une calculatrice en AngularJS

Utiliser le 1-way binding et la gestion simple d'événements